CS189: Introduction to Machine Learning

Homework 3

Due: February 25, 2014, 11:59 pm

Problem 1: Linear Regression for Polynomial Curve Fitting

Suppose that we are interested in fitting polynomials to data; in particular, consider the polynomial regression model linking the response variable $y \in \mathbb{R}$ to the covariate $x \in \mathbb{R}$ via

$$y = \sum_{k=0}^{d} \beta_k x^k + w \tag{1}$$

where $w \sim \mathcal{N}(0, 1)$ is Gaussian noise.

(a) The files `Ytrain.txt` and `Xtrain.txt` contain samples $\{x_i, y_i\}_{i=1}^n$, with $n = 100$. For $d = 1, 2, \ldots, 10$, fit the above model to this data by minimizing the least squares loss

$$L(\beta) = \sum_{i=1}^{n} \left(y_i - \sum_{k=0}^{d} \beta_k (x_i)^k \right)^2$$

For which choice of d is this cost function smallest? Is this the best choice for d? Why/why not? On the same figure, plot the original data and the models to the data for $d = 1, 2, 3, 4$.

(b) Given the polynomial coefficients $\hat{\beta}[d]$ for a particular choice of d and a new covariate x, one can generate a predicted response $\hat{y}$ as

$$\hat{y} = \sum_{k=0}^{d} \hat{\beta}_k x^k$$

The files `Ytest.txt` and `Xtest.txt` contain $m = 500$ new samples. Using the samples in `Xtest.txt`, generate predictions $\hat{y}_i$ for $i = 1, \ldots, m$, and compute the prediction error $\sum_{i=1}^{m} (\hat{y}_i - y_i)^2$ for $d = 3$ and $d = 10$. What can you infer from these?

Problem 2: Isocontours of Normal distributions

Let $f(\mu, \Sigma)$ denote the density function of a Gaussian random variable. Plot isocontours of the following functions:

i) $f(\mu, \Sigma)$, where $\mu = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$

ii) $f(\mu, \Sigma)$, where $\mu = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$

iii) $f(\mu_1, \Sigma_1) - f(\mu_2, \Sigma_2)$, where $\mu_1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$, $\mu_2 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ and $\Sigma_1 = \Sigma_2 = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$

iv) $f(\mu_1, \Sigma_1) - f(\mu_2, \Sigma_2)$, where $\mu_1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$, $\mu_2 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$, $\Sigma_1 = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$ and $\Sigma_2 = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$

v) $f(\mu_1, \Sigma_1) - f(\mu_2, \Sigma_2)$, where $\mu_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\mu_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$, $\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$ and $\Sigma_2 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$

Problem 3: Gaussian classifiers for digits

In this problem we will build Gaussian classifiers for digits in MNIST. More specifically, we will model each digit class as a Gaussian distribution and make our decisions on the basis of posterior probabilities. This is a generative method for classifying images where we are modelling the class conditional probabilities as normal distributions. The steps mentioned below should be done for each training set in `train_small.mat` and you need to plot a curve of error rate vs no. of training examples upon evaluating on the test set in `test.mat`.

i) Taking raw pixel values as features, fit a Gaussian distribution to each digit class using maximum likelihood estimation. This involves finding the means and covariance matrices for each digit class. What are the maximum likelihood estimates for the mean and covariance matrix of a Gaussian distribution? Are these estimators unbiased?
Tip: It is a good idea to contrast normalize images before using the raw pixel values. One way of normalization is to divide the pixel values of an image by the l_2 norm of its pixel values.

ii) How would you model the prior distribution for each class? Compute prior probabilities for all classes.

iii) Visualize the covariance matrix for a particular class. Do you see any kind of structure in the matrix? What does this symbolize?

iv) We will now classify digits in the test set on the basis of posterior probabilities using two different approaches:

a) Define $\Sigma_{overall}$ to be the average of the covariance matrices of all the classes. We will use this matrix as an estimate of the covariance of all the classes. This amounts to modelling class conditionals as Gaussians ($\sim \mathcal{N}(\mu_i, \Sigma_{overall})$) with different means and the same covariance matrix. Using this form of class conditional probabilities, classify the images in the test set into one of the 10 classes assuming 0-1 loss and compute the error rate. What is the form of the decision boundary in this case? Why?

b) We can also model class conditionals as $\mathcal{N}(\mu_i, \Sigma_i)$, where Σ_i is the estimated covariance matrix for the i^{th} class. Classify images in the test set using this form of the conditional probability (assuming 0-1 loss) and compute the error rate. What is the form of the decision boundary in this case?

c) Compare your results in parts *a* and *b*. What do you think is the source of difference in the performance?

Note: The covariance matrices you compute using MLE might be singular (and thus non-invertible). In order to make them non-singular and positive definite, you can add a small weight to their diagonals by setting $\Sigma_i = \Sigma_i + \alpha I$, where α is the weight you want to add to the diagonals.

In your submission, you need to include learning curves (error-rate vs no. of training examples) and actual error-rate values for the above two cases and short explanations for the all the questions.

Problem 4: Centering and ridge regression

You are given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$. Let $\mathbf{X}$ be the design matrix (i.e. the matrix whose i^{th} row is $\mathbf{x}_i$), and let $\mathbf{y}$ be the column vector whose i^{th} entry is y_i. Let $\mathbf{1}$ be a $n \times 1$ column vector of ones.

Define $\bar{\mathbf{x}} = \frac{1}{n}\sum_i \mathbf{x}_i$ and $\bar{y} = \frac{1}{n}\sum_i y_i$. Assume that the input data has been centered, so that $\bar{\mathbf{x}} = 0$. Show that the optimizer of

$$J(\mathbf{w}, w_0) = (\mathbf{y} - \mathbf{X}\mathbf{w} - w_0\mathbf{1})^\top(\mathbf{y} - \mathbf{X}\mathbf{w} - w_0\mathbf{1}) + \lambda \mathbf{w}^\top \mathbf{w}$$

is given by

$$\begin{aligned} \hat{w_0} &= \bar{y} \\ \hat{\mathbf{w}} &= (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}. \end{aligned}$$

Problem 5: MLE for simple linear regression

Simple linear regression refers to the case of linear regression in which the input is a scalar quantity.

Let the data set be $\{(x_i, y_i)\}_{i=1}^n$, where each sample is drawn independently from a joint distribution over input and output: $(x_i, y_i) \sim (X, Y)$. Assume the Gaussian noise setting:

$$y_i | x_i \sim \mathcal{N}(w_0 + w_1 x_i, \sigma^2)$$

Show that the MLE in this simple linear regression model is given by the following equations, which may be familiar from basic statistics classes:

$$w_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = \frac{\sum_i x_i y_i - n\bar{x}\bar{y}}{\sum_i x_i^2 - n\bar{x}^2} \approx \frac{\text{cov}(X, Y)}{\text{var}(X)}$$

$$w_0 = \bar{y} - w_1 \bar{x} \approx \text{E}[Y] - w_1 \text{E}[X]$$

Submission Instructions

In your submission, you need to include a write up with answers to all the questions and the plots. You also need to include your code and a README with instructions as to how we can run your code. All solutions should be submitted via bSpace.

As always, you can work with one partner. If you choose to do so, please indicate in the README and in your bSpace submission the names and student IDs of yourself and your partner. Only one of you needs to submit the assignment.

Good luck!